UNIVERSITY OF TWENTE.

Formal Methods & Tools.

Model-based Testing with Graph Grammars

Vincent de Bruijn

September 4th, 2012

M.Sc. Colloquium

Model-based Testing (1/3)

- Why testing?

Model-based Testing (1/3)

- Why testing?
    - List of requirements

Model-based Testing (1/3)

- Why testing?
  - List of requirements
  - Test if implementation satisfies requirements

Model-based Testing (1/3)

- Why testing?
    - List of requirements
    - Test if implementation satisfies requirements
- Creating tests manually:

Model-based Testing (1/3)

- Why testing?
  - List of requirements
  - Test if implementation satisfies requirements
- Creating tests manually:
  - Error-prone

Model-based Testing (1/3)

- Why testing?
  - List of requirements
  - Test if implementation satisfies requirements
- Creating tests manually:
  - Error-prone
  - Time intensive

Model-based Testing (1/3)

- Why testing?
  - List of requirements
  - Test if implementation satisfies requirements
- Creating tests manually:
  - Error-prone
  - Time intensive
- Solution

Model-based Testing (1/3)

- Why testing?
    - List of requirements
    - Test if implementation satisfies requirements
- Creating tests manually:
    - Error-prone
    - Time intensive
- Solution
    - Create model from the requirements

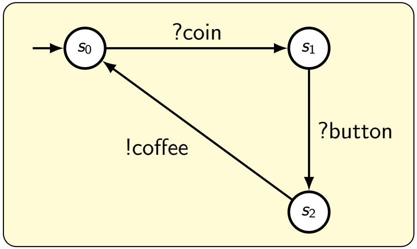
Model-based Testing (1/3)

- Why testing?
  - List of requirements
  - Test if implementation satisfies requirements
- Creating tests manually:
  - Error-prone
  - Time intensive
- Solution
  - Create model from the requirements
  - Generate tests automatically using model

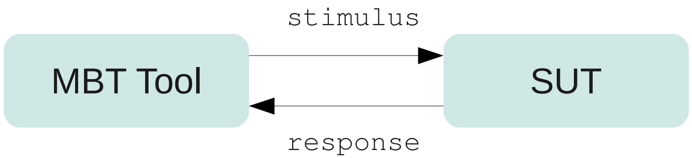
Model-based Testing (2/3)

Model

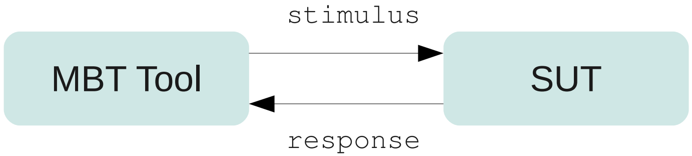
- An abstract representation of the behavior of a system



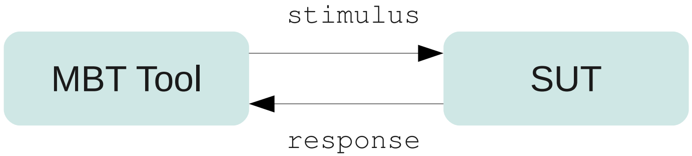
Model-based Testing (3/3)



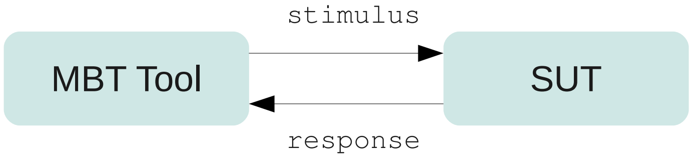
Model-based Testing (3/3)



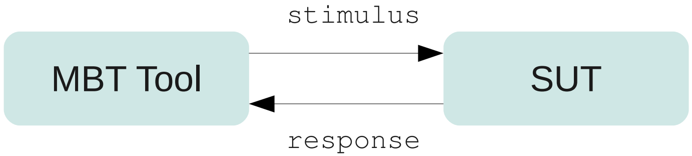
Model-based Testing (3/3)



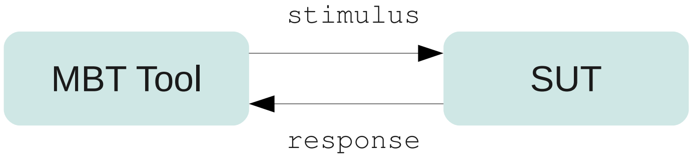
Model-based Testing (3/3)



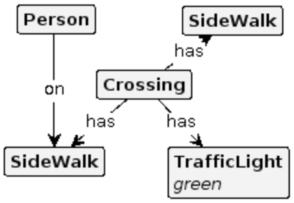
Model-based Testing (3/3)



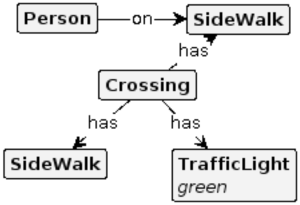
Model-based Testing (3/3)



Graph Grammars (1/2)



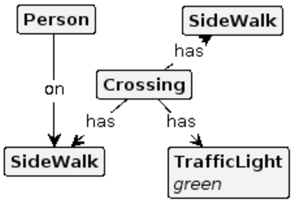
before



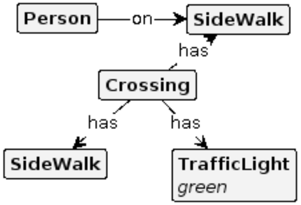
after

- Graphs represent system states

Graph Grammars (1/2)



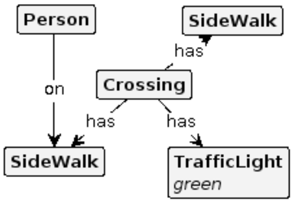
before



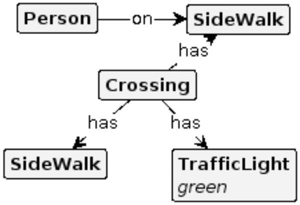
after

- Graphs represent system states
- Graph rules express possible changes to graph

Graph Grammars (1/2)



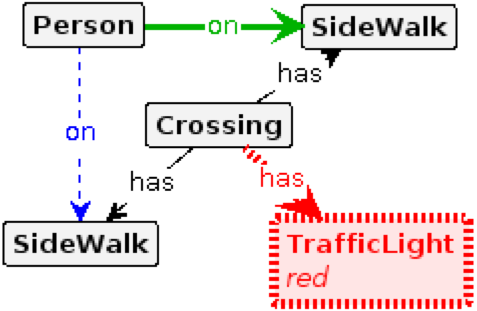
before



after

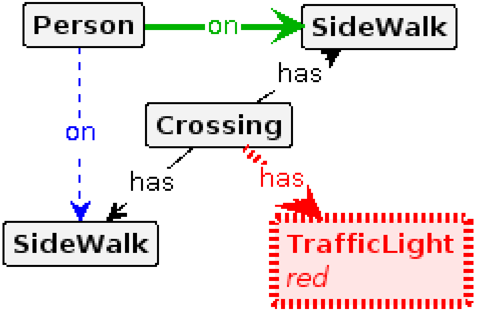
- Graphs represent system states
- Graph rules express possible changes to graph
- All possible changes make a *Graph Transition System*

Graph Grammars (2/2)



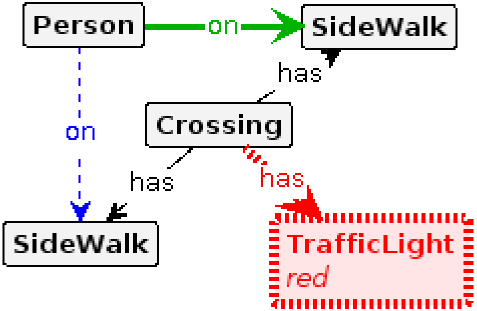
- Black and blue parts have to be present in graph

Graph Grammars (2/2)



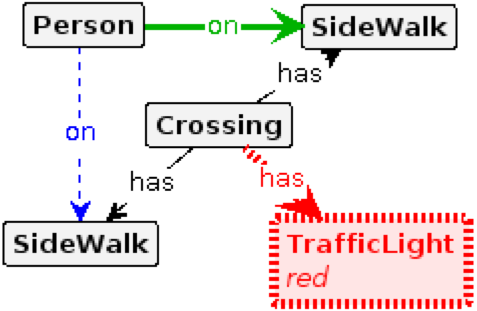
- Black and blue parts have to be present in graph
- Red parts may not be present in graph

Graph Grammars (2/2)



- Black and blue parts have to be present in graph
- Red parts may not be present in graph
- Blue is erased from graph

Graph Grammars (2/2)



- Black and blue parts have to be present in graph
- Red parts may not be present in graph
- Blue is erased from graph
- Green is added to graph

Tools

- Axini Test Manager (ATM)

Tools

- Axini Test Manager (ATM)
- GRaphs for Object-Oriented VErification (GROOVE)

Research Goals

- Goals

Research Goals

- Goals
  - Use GROOVE and ATM to create model-based testing tool with Graph Grammars

Research Goals

- Goals
  - Use GROOVE and ATM to create model-based testing tool with Graph Grammars
  - Validate this tool using case studies

Research Goals

- Goals
  - Use GROOVE and ATM to create model-based testing tool with Graph Grammars
  - Validate this tool using case studies
- Motivation

Research Goals

- Goals
  - Use GROOVE and ATM to create model-based testing tool with Graph Grammars
  - Validate this tool using case studies
- Motivation
  - Graphs are well-known and often used to represent system states

Research Goals

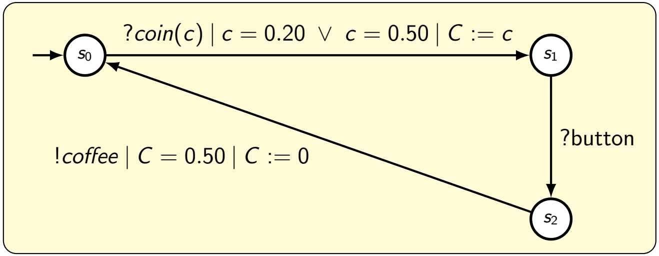
- Goals
  - Use GROOVE and ATM to create model-based testing tool with Graph Grammars
  - Validate this tool using case studies
- Motivation
  - Graphs are well-known and often used to represent system states
  - Rules are useful for describing computations

Inhoudsopgave

Inhoudsopgave

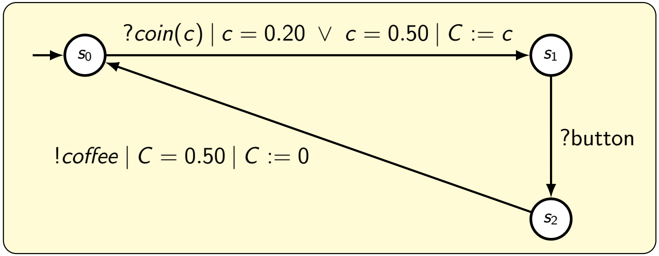
Setup (1/2)

- Graphs for humans, transition systems for computers


$$\begin{tikzcd}
\arrow[r] & s_0 \arrow[rr, "?coin(c) \mid c = 0.20 \vee c = 0.50 \mid C := c"] & & s_1 \arrow[d, "?button"] \\
& & & s_2 \arrow[ull, "!coffee \mid C = 0.50 \mid C := 0"]
\end{tikzcd}$$

Setup (1/2)

- Graphs for humans, transition systems for computers
- ATM uses *Symbolic Transition Systems*



$$\begin{tikzcd} \arrow[r] & s_0 \arrow[rr, "?coin(c) \mid c = 0.20 \vee c = 0.50 \mid C := c"] & & s_1 \arrow[d, "?button"] \\ & & & s_2 \arrow[ull, "!coffee \mid C = 0.50 \mid C := 0"] \end{tikzcd}$$

Setup (2/2)

- The tool:

Setup (2/2)

- The tool:
  1. creates STS from the GG in GROOVE

Setup (2/2)

- The tool:
  1. creates STS from the GG in GROOVE
  2. sends STS to ATM

Setup (2/2)

- The tool:
  1. creates STS from the GG in GROOVE
  2. sends STS to ATM
  3. does model-based testing in ATM

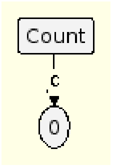
Setup (2/2)

- The tool:
  1. creates STS from the GG in GROOVE
  2. sends STS to ATM
  3. does model-based testing in ATM
- Step number 1 main part of this research.

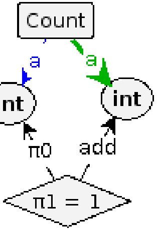
Inhoudsopgave

Algorithm

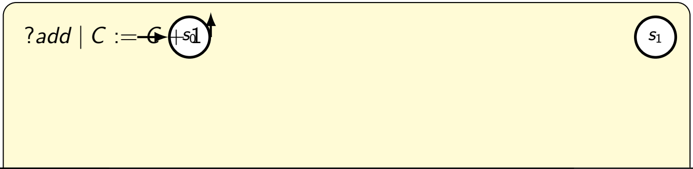
1. Create variables from data values



graph

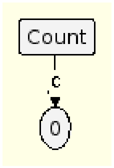


rule

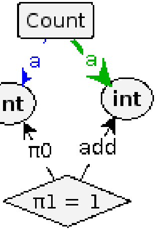


Algorithm

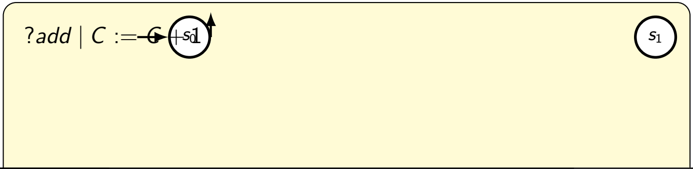
1. Create variables from data values
2. Explore GTS disregarding data values



graph

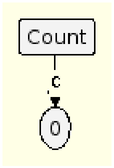


rule

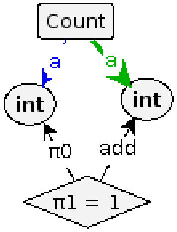


Algorithm

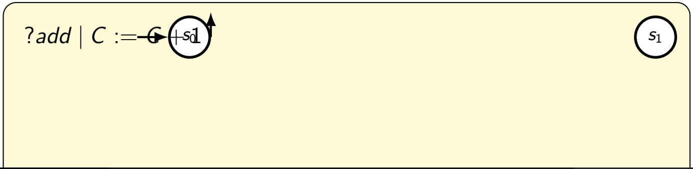
1. Create variables from data values
2. Explore GTS disregarding data values
3. Parse guards and updates from rules



graph



rule



Constraints

1. Variables have to be unique

one picture here with all mistakes.

Constraints

1. Variables have to be unique
2. Variables cannot be part of NACs

one picture here with all mistakes.

Constraints

1. Variables have to be unique
2. Variables cannot be part of NACs
3. Structural constraints on node creating rules

one picture here with all mistakes.

Inhoudsopgave

Model Examples

- 4 small examples used:

Model Examples

- 4 small examples used:
  1. a boardgame

Model Examples

- 4 small examples used:
  1. a boardgame
  2. a puzzle

Model Examples

- 4 small examples used:
  1. a boardgame
  2. a puzzle
  3. a reservation system

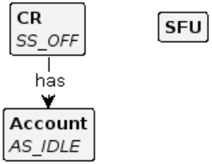
Model Examples

- 4 small examples used:
  1. a boardgame
  2. a puzzle
  3. a reservation system
  4. a bar tab system

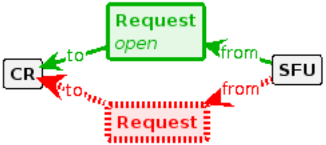
Case study (1/2)

- Self-checkout register

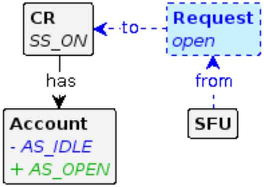
Case study (2/2)



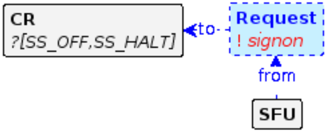
graph



request



response



error

Inhoudsopgave

Conclusion

- Created a tool for model-based testing with Graph Grammars

Conclusion

- Created a tool for model-based testing with Graph Grammars
- Transformation needs to be extended: complex data structures

Conclusion

- Created a tool for model-based testing with Graph Grammars
- Transformation needs to be extended: complex data structures
- Modelling behavior with GGs is effective